交通工程学
Introduction to Traffic Engineering
第 14 节　TSP 与启发式算法[1]

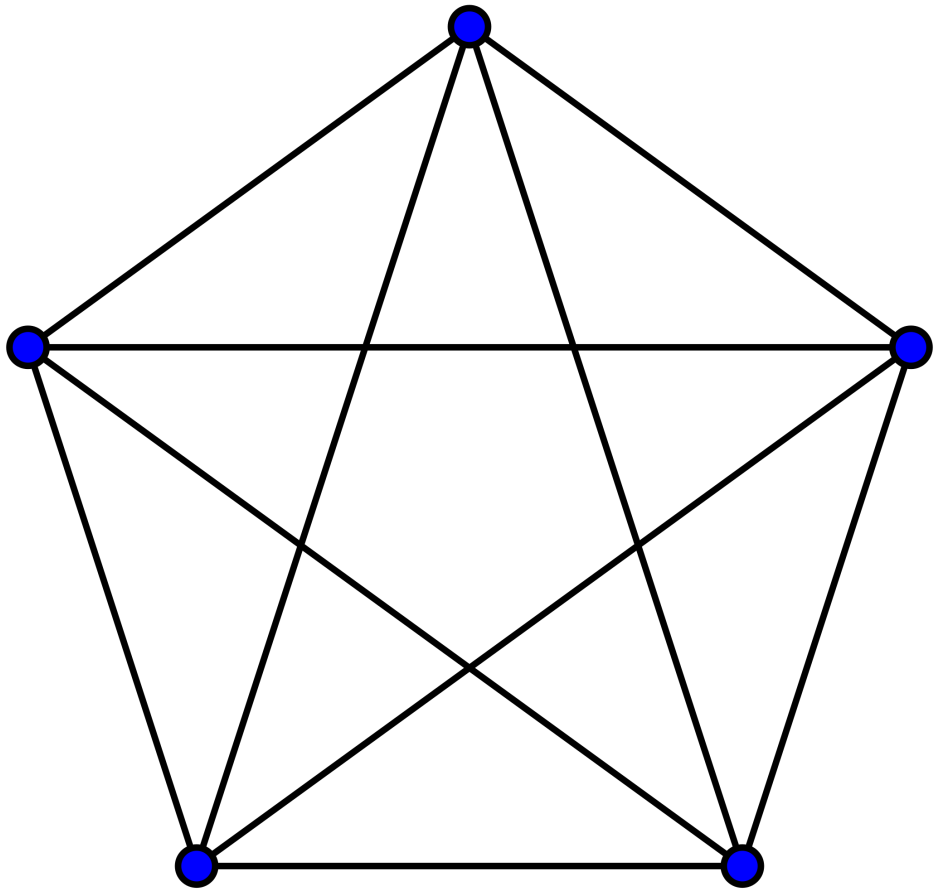
葛乾

西南交通大学　系统科学与系统工程研究所
西南交通大学　交通工程系

[1]基于 Anil Aswani, Misha Lavrov, Martin Savelsbergh, Wendy Williams 等人讲义。仅供教学使用，请勿传播。

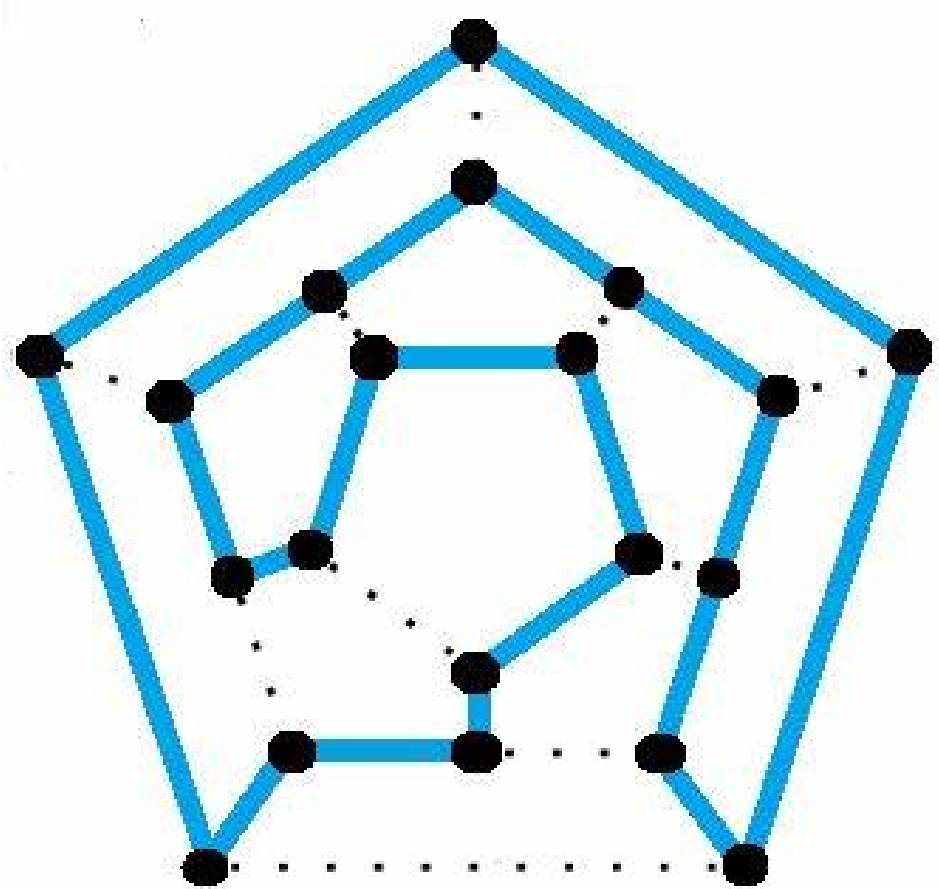
本节目录

完全图 (complete graph)



- 无向图
- 任意两个节点都有弧段相连

哈密顿回路 (Hamilton cycle)



- 从图中某个节点出发，遍历其他节点各一次后回到起点的回路

本节目录

TSP 问题的描述

给定以下条件

- 一组有待访问的地点 $1, 2, 3, \ldots, n$（对应途中的节点）
- 任意两个地点 i, j 之间的旅行费用 c_{ij}

目标：寻找连接所有地点的哈密顿回路，使得总费用最小
通常假设

- $c_{ij} = c_{ji}$，节点之间的旅行费用是对称的，方向不影响其费用
- $c_{ij} + c_{jk} \geqslant c_{ik}$，三角不等式

如何建模?

- 用 $x_{ij} \in \{0, 1\}$ 描述 i 与 j 之间是否存在一个旅途，$x_{ij} = 1$ 表示存在，$x_{ij} = 0$ 表示不存在
- 最小化所有旅途的总费用

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

- 进入每个地点一次

$$\sum_{1 \leqslant i \leqslant n, i \neq j} x_{ij} = 1 \qquad \forall\, j = 1, 2, \ldots, n$$

- 离开每个地点一次

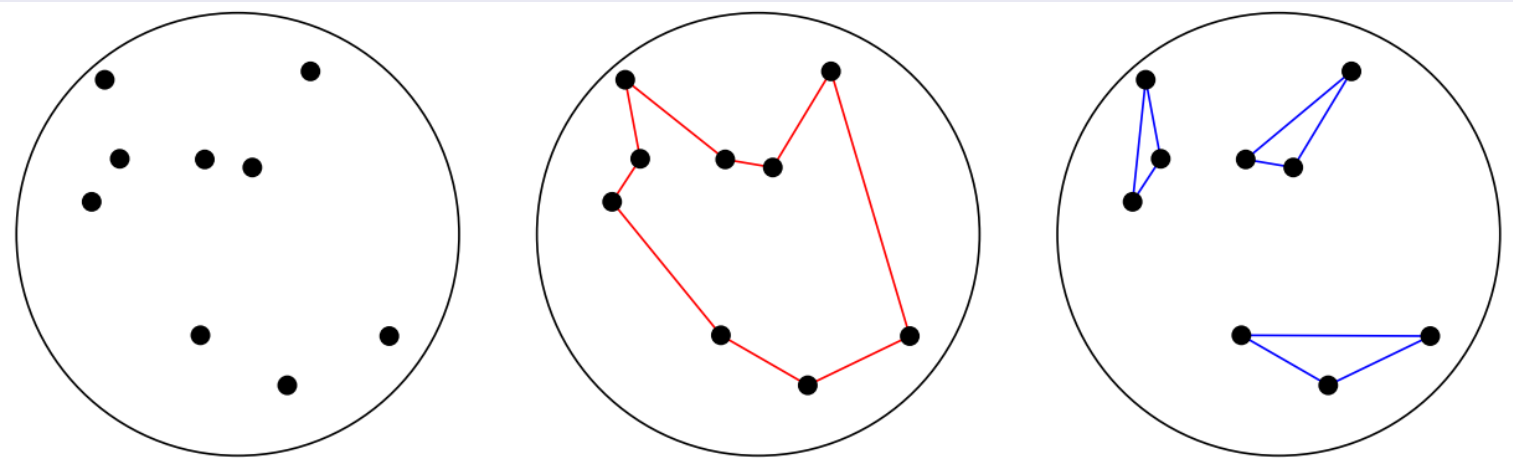
$$\sum_{1 \leqslant k \leqslant n, k \neq j} x_{jk} = 1 \qquad \forall\, j = 1, 2, \ldots, n$$

问题–子路径

- 上述的建模方式不能保证我们刚好通过一次旅行就访问了 n 个地点

示例

假设有 9 个地点需要访问，地点之间的旅行费用为二者的欧拉距离。则最优的旅行为中图红色所示，而我们之前所求出的最优解则为右图蓝色所示。右图中蓝色的子路径有三个，彼此互不相连。



子路径消除约束（subtour elimination constraints，SEC）

DFJ 解决方法

对于所有的地点的任一子集 S，则生成的旅行至少离开 S，前往剩余节点一次（Dantzig, Fulkerson, Johnson, 1954）

- For every $S \subset \{1, 2, \ldots, n\}$ with $1 \leqslant |S| \leqslant n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geqslant 1$$

- 该规则适用于任何子路径，最终，路径将连通 S 中的节点与不在 S 中的节点

优劣

- 优: 子路径消除约束与原先的约束，共同可以刚好生成包含所有节点的路径
- 劣: 对于包含 n 个节点的问题，有 $2^{n-1}-2$ 个子路径消除约束！($2^{n-1}-1$ 如果假设起始节点 $1 \in S$.)
- 稍微好点的信息: 如果给我们一个不考虑 SEC 的 TSP 问题的解，我们可以很容易验证，该解不满足哪一个 SEC。例如，令 S 表示从节点 1 开始，被某个路径所访问的所有节点集合
  - 如果 $S = \{1, 2, \ldots, n\}$，则刚好得到我们想要的路径
  - 否则，要求至少离开 S 一次的约束条件被违反了

本节目录

分支切割法（思路）

- 用 SEC 可以构造一个分支切割算法，用来求解 TSP 问题
- 我们可以从 TSP 问题（不含 SEC）的线性约束出发，生成一个解
  - 假设解中存在不是整数的解 $x_{ij} \notin Z$，在此处分支 $x_{ij} = 0$，$x_{ij} = 1$
  - 假设这个解是一个存在子路径的整数解，则找到该子路径违反的 SEC，把子路径的约束加进问题中
  - 假设这个解是一个完整路径的整数解，则对比该解与现在的下界，更新最好的解

子路径消除约束（subtour elimination constraints，SEC）

MTZ 解决方法

添加一组新变量 t_i，表示访问各个地点的时间，则 $1 \leqslant t_i \leqslant n-1$，同时 t_1 不受约束（因为最终还要回到该点）（Miller, Tucker, Zemlin, 1960）
假设我们的路径中存在 x_{ij} 这一个值，则表明 i 先于 j 被访问，即对于任意一组节点 $i, j \neq 1$ 存在以下逻辑约束

$$\text{if } x_{ij} = 1, \text{ then } t_j \geqslant t_i + 1$$

为什么该方法有效?

- **对于包含所有节点的路径，其满足 MTZ 约束**。假设我们访问的地点的顺序是 $\{i_1, i_2, \ldots, i_{n-1}\}$，则 $t_{i_1} = 1, t_{i_2} = 2, \ldots, t_{i_1} = n - 1$
- **假设求得的路径为子路径，则其无法满足 MTZ 约束**。假设我们的子路径是 $\{a, b, c\}$，即 $x_{ab} = x_{bc} = x_{ca} = 1$，同时 $a, b, c \neq 1$。MTZ 约束要求其满足

$$t_b \geqslant t_a + 1$$

$$t_c \geqslant t_b + 1$$

$$t_a \geqslant t_c + 1$$

易知该条件无法被满足

逻辑约束的处理

- MTZ 约束：对于任意一组节点 $i, j \neq 1$ 存在以下逻辑约束

$$\text{if } x_{ij} = 1, \text{ then } t_j \geqslant t_i + 1$$

- 其等价于：对于足够大的 M

$$t_j \geqslant t_i + 1 - M(1 - x_{ij})$$

  - $x_{ij} = 1 \rightarrow t_j \geqslant t_i + 1$
  - $x_{ij} = 0 \rightarrow t_j \geqslant t_i + 1 - M$，因为 M 足够大，因而对 t_i, t_j 无限制
- 可以验证 $M = n$ 符合我们要求。$t_1, t_2, \ldots, t_n$ 可以从 $1, 2, \ldots, n-1$ 中取值，因此 $t_j \geqslant t_i + 1 - n$ 永远成立

两种方法对比

建模上看

- DFJ 方法增加了 $2^{n-1}-1$ 个约束
- MTZ 方法仅增加 n^2 个约束，以及 $n-1$ 个变量 t_i，这些变量的取值可以为整数，但并不一定是

实际的求解

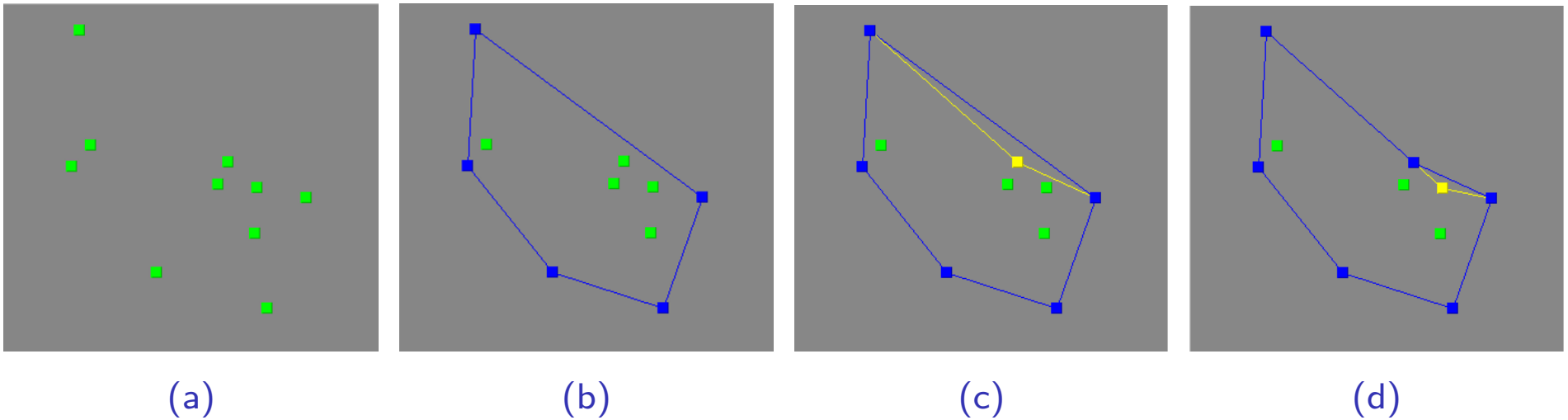
- DFJ 方法可以使用分支切割法求解
- MTZ 方法引入了大 M，而 M 的存在，增大了线性松弛问题的解空间。用分支定界法求解时反而更难收敛

本节目录

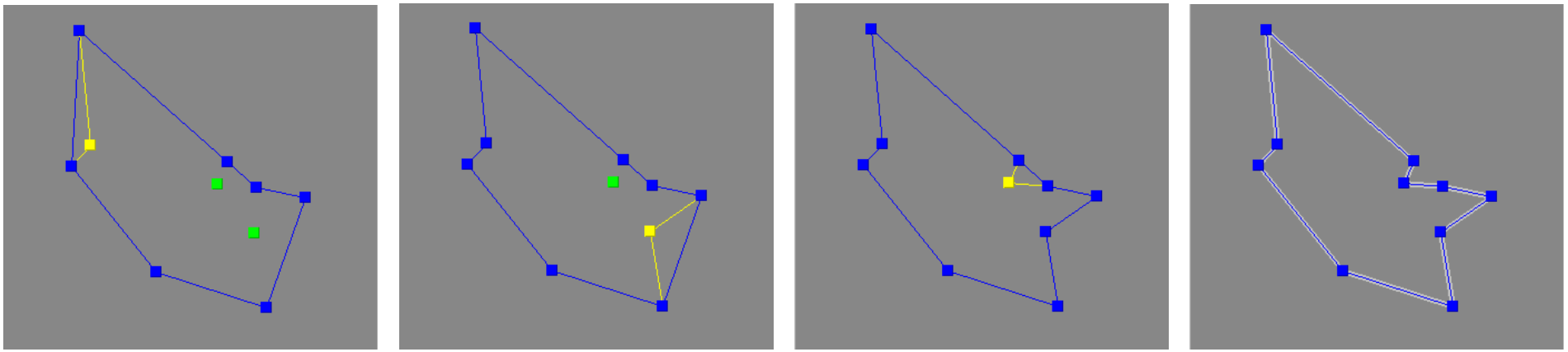
插入算法 (insertion heuristics)

- 插入算法先从一个子路径出发，不断地逐个添加剩余的节点，将其插入到成本最小的位置，直到所有节点均在路径中
  - 最小成本规则：每次添加离子路径最近的点
  - 最远距离规则：每次添加离子路径最远的点

最小成本规则



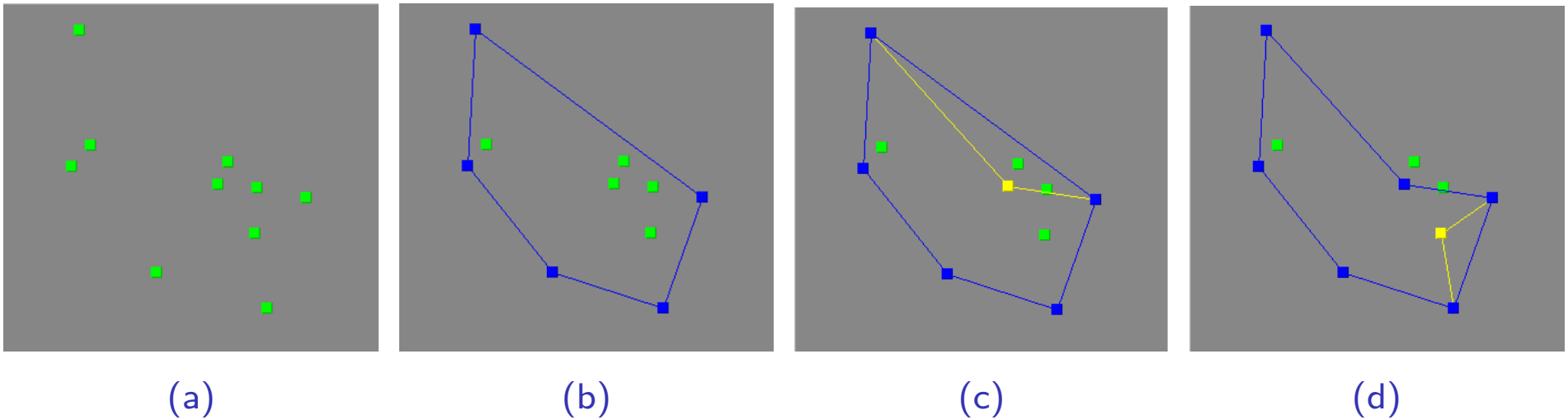
(a) (b) (c) (d)



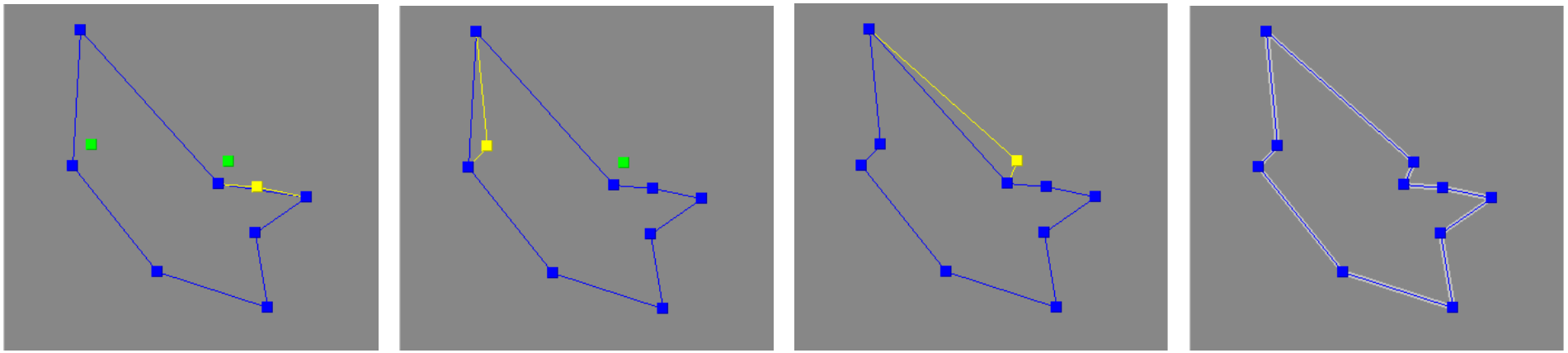
(e) (f) (g) (h)

Figure

最远距离规则



(a) (b) (c) (d)



(e) (f) (g) (h)

Figure

插入算法注意点

- 开始的子路径为所有节点的凸包（convex hull）
- 两种规则的优点是，避免了路径中产生相交的线段。而出现相交线段，则意味着解不是最优
- 用启发式规则可以获得可行解，但是解是否最优无法证明
- 迄今为止，尚不存在既能有效，也能精确求解 TSP 的问题的方法

2-opt

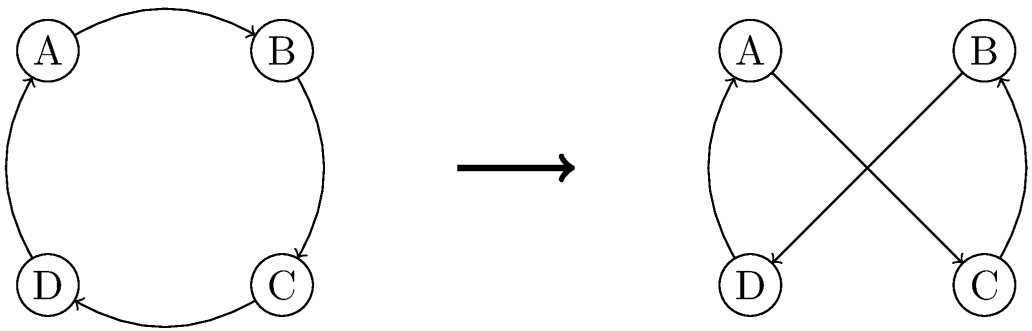


Figure: (A,D) 与 (B,C) 为两个子路径

3-opt

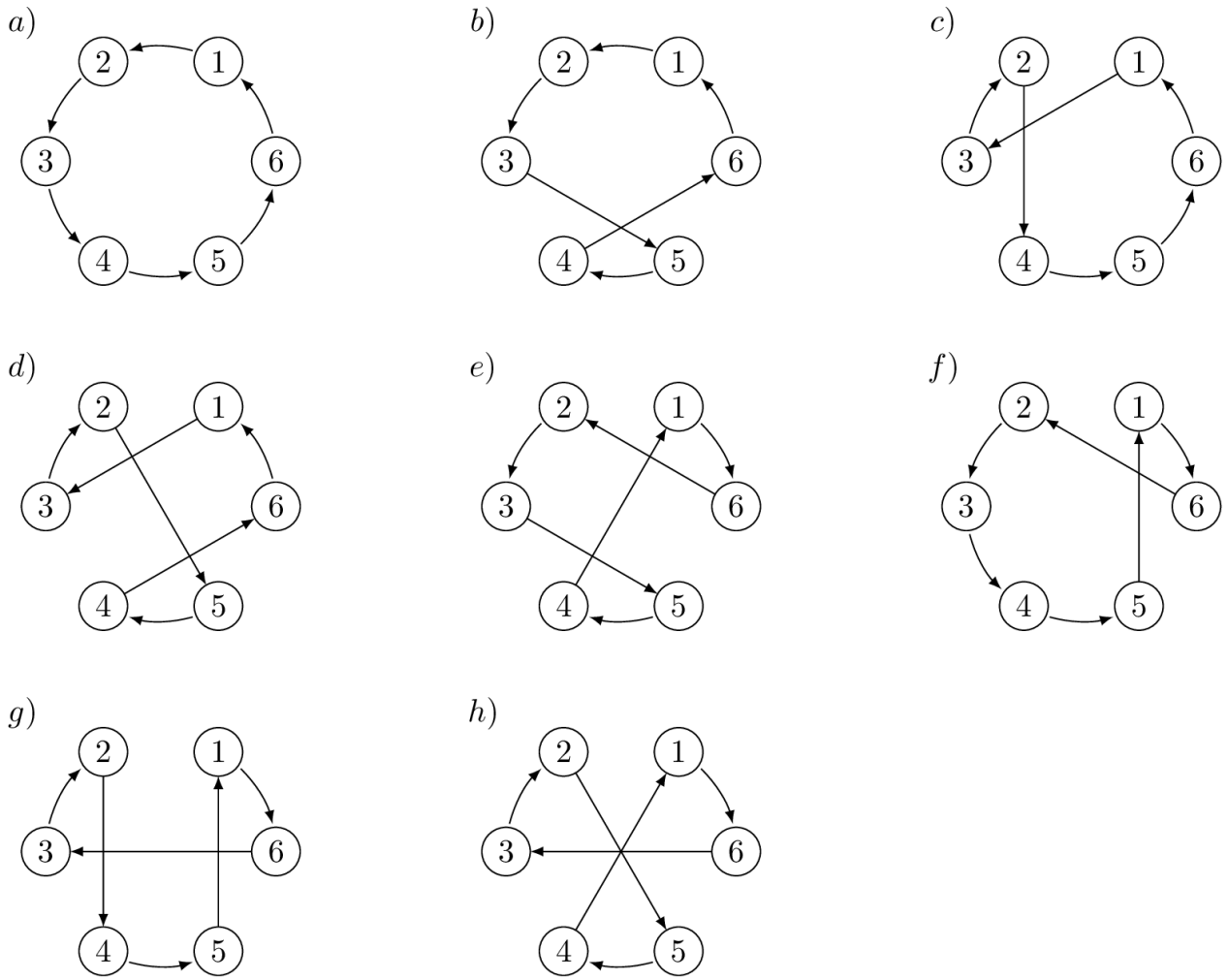


Figure: (1,6)、(2,3)、(4,5) 为三个子路径

其他启发式规则

- Lin-Kernighan 算法
- 分支定界算法
- ...

本节目录

元启发式算法

- The term metaheuristics was coined by Glover in 1986 and in general means 'to find beyond in an upper level.'
- Metaheuristics include Tabu Search (禁忌搜索), Simulated Annealing (模拟退火), Ant Colony Optimization (蚁群优化), Evolutionary Computation (进化计算), iterated local search (迭代本地搜索), and Memetic Algorithms (Memetic 算法) 等
- Metaheuristics do not guarantee that near-optimal solutions will be found quickly for all problem instances. However, these complex programs do find nearoptimal solutions for many problem instances that arise in practice.
  - 为什么学习元启发式算法？在现实生活中，改进一个解，有时比寻找最优解更重要
- 我们仅介绍进化计算中的**遗传算法**

遗传算法

- Directed search algorithms based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970's)
  - To understand the adaptive processes of natural systems
  - To design artificial systems software that retains the robustness of natural systems

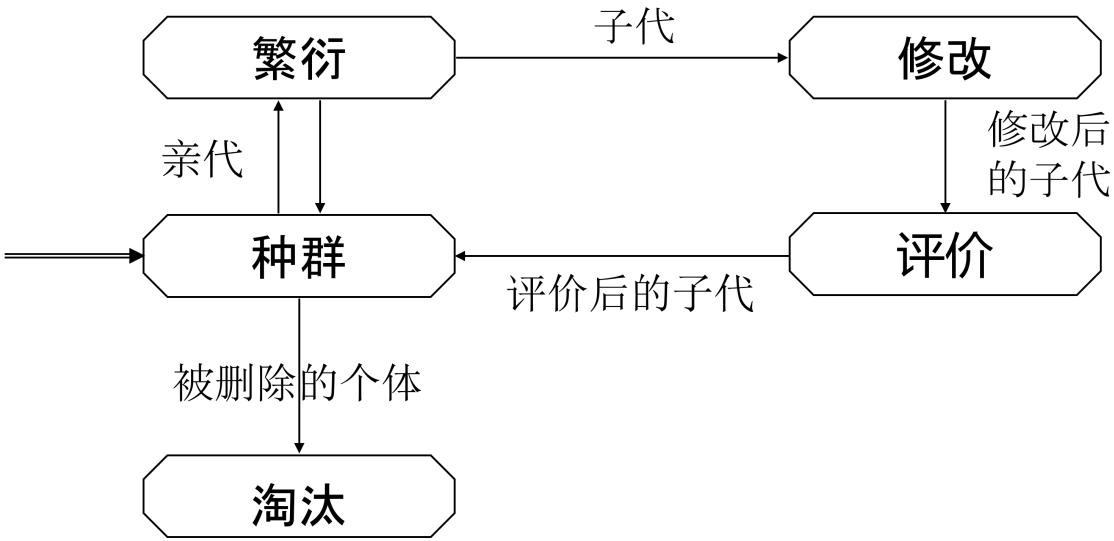
遗传算法的构成要素

- Encoding technique（编码规则）(gene, chromosome)
- Initialization procedure（初始化过程）(creation)
- Evaluation function（评价函数）(environment)
- Selection of parents（亲代的选择）(reproduction)
- Genetic operators（基因操作）(mutation, recombination)
- Parameter settings（参数设定）(practice and art)

整体算法

- 初始化种群（可行解）;
- 评价种群;
- while 未达到终止条件
  1. 从种群中选择繁衍的亲代;
  2. 基因重组和变异;
  3. 评价种群;

流程图



种群

种群的染色体可以为:

- 二进制字符串 (0101 ... 1100)
- 实数 (43.2 -33.1 ... 0.0 89.2)
- 元素的排序 (E11 E3 E7 ... E1 E15)
- 规则的集合 (R1 R2 R3 ... R22 R23)
- 编程元素 (genetic programming)
- 其他任意数据结构

繁衍

- 根据染色体编码规则，从种群中随机选取

染色体修改

- 染色体的修改是随机触发的，包括变异与重组两种修改
- 变异

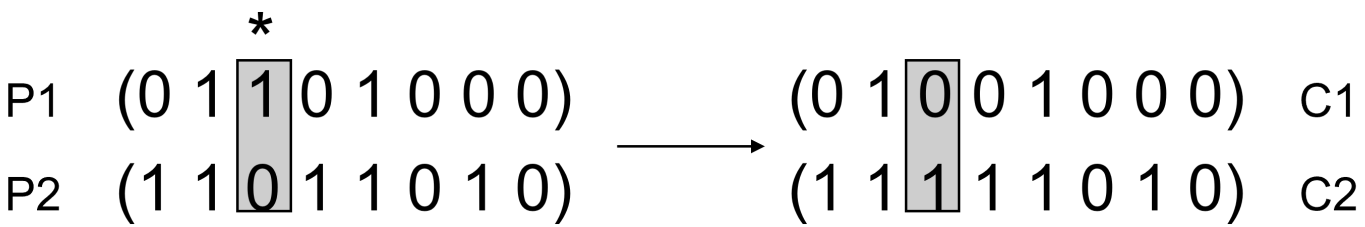
Before: (1 0 1 **1** 0 1 1 0)

After: (1 0 1 **0** 0 1 1 0)

Before: (1.38 **-69.4** 326.44 0.1)

After: (1.38 **-67.5** 326.44 0.1)

- 重组



评价与淘汰

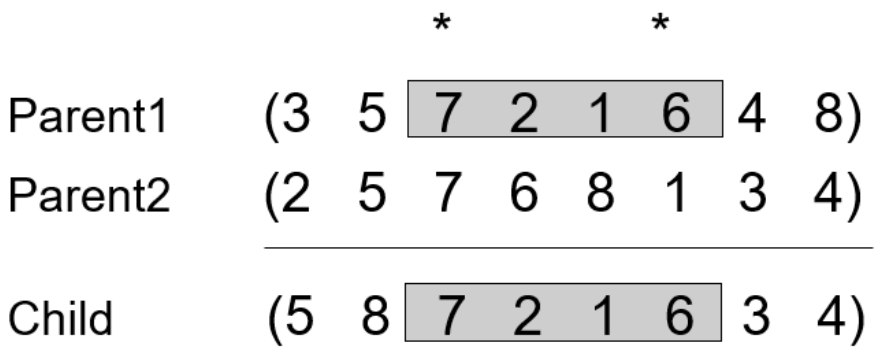
- 在评价步骤中我们对染色体进行解码（转换成原先格式的数据），并衡量其适应度。评价是遗传算法与原问题唯有的联系
- 在淘汰这一步，可以将所有的种群全部换成新生的子代，也可进替换部分

如何用到 TSP 问题上?

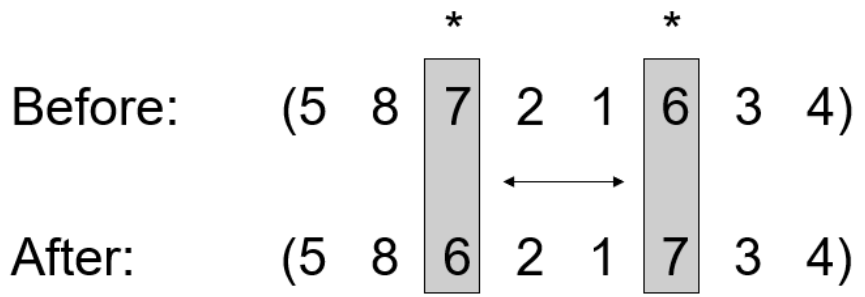
- 假设我们有 8 个城市待访问，1) London、2) Venice、3) Dunedin、4) Singapore、5) Beijing、6) Phoenix、7) Tokyo、8) Victoria
- Representation is an ordered list of city numbers known as an order-based GA
- 表示方式可以是
  - CityList1 (3 5 7 2 1 6 4 8)
  - CityList2 (2 5 7 6 8 1 3 4)

染色体修改

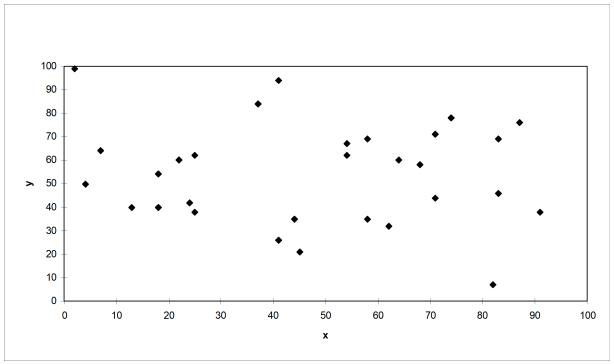
- order1 **重组**法：从亲代中找到起始点，将其中一个亲代的起始点之间的遗传片段复制给子代；从起始点开始，将第二个亲代中不重复的数据复制给子代



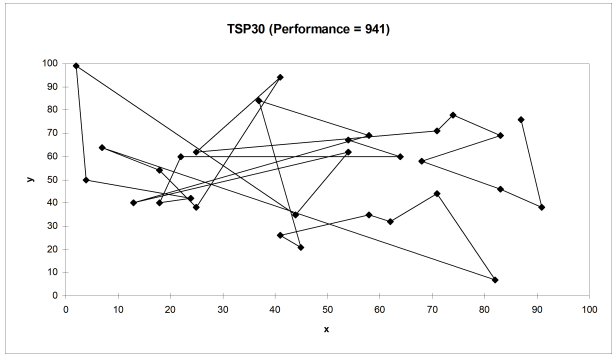
- 变异：交换链表中两个点的位置



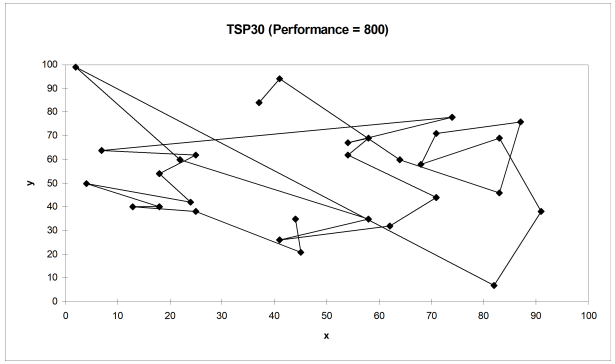
TSP 问题求解示例



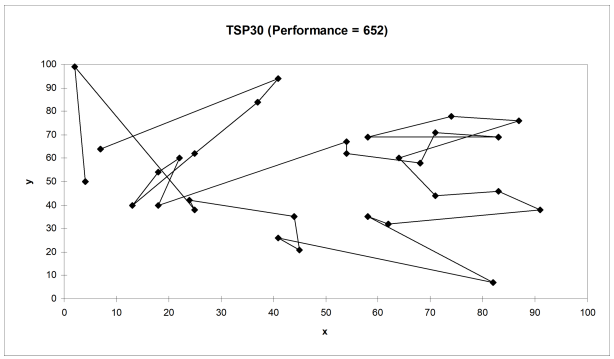
(a) 30 个城市



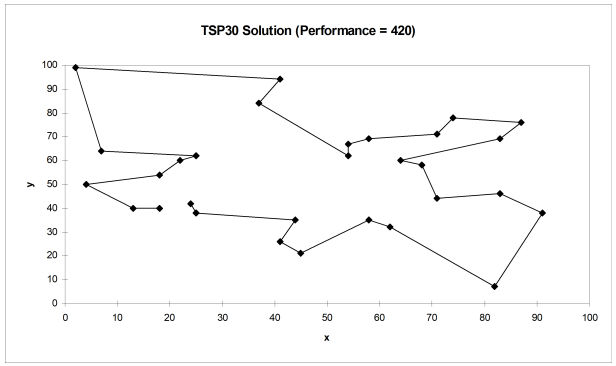
(b) Distance = 941



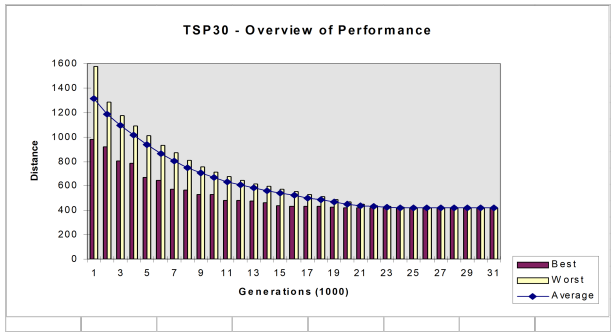
(c) Distance = 800



(d) Distance = 652



(e) Distance = 420



(f) 算法表现

谢谢!